Processing a Delineated Watershed to Include Green & Ampt Soil Parameters

by

Jerome Richard Perales, B.S., B.A.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2000

Processing a Delineated Watershed to Include Green & Ampt Soil Parameters

Approved by
Supervising Committee:

David R. Maidment

Francisco Olivera

Dedication

To Mom and Dad

Chapter 1 Introduction

Infiltration is the movement of water downward through the soil. The rate at which water can flow through the soil depends on the properties of the particular soil and it's water content at the surface. This loss of water to the soil is known as an abstraction. Other loss mechanisms include streambed losses, interception, and storage within topographical depressions. There are several models to calculate abstraction due to infiltration including the SCS method and the initial loss plus constant loss rate model. However, these models have limitations. The SCS Curve Numbers relating abstractions to soil types and land use have been arbitrarily determined, and the initial loss plus constant loss rate model is a non-physical model calibrated with gauged data. A more advantageous method to determine abstractions is to relate loss rates to physical characteristics of the watershed in a continuous simulation so that loss rates may be computed as a function of soil moisture conditions at the beginning of a storm (Reed and Maidment 1998).

The Green & Ampt Infiltration Model is a physical model which relates the rate of infiltration to measurable soil properties such as the porosity, hydraulic conductivity, and the moisture content of a particular soil. An illustration of simplified model proposed by Green and Ampt is shown in Fig. 1 (Source: Chow, Maidment, and Mays 1988).

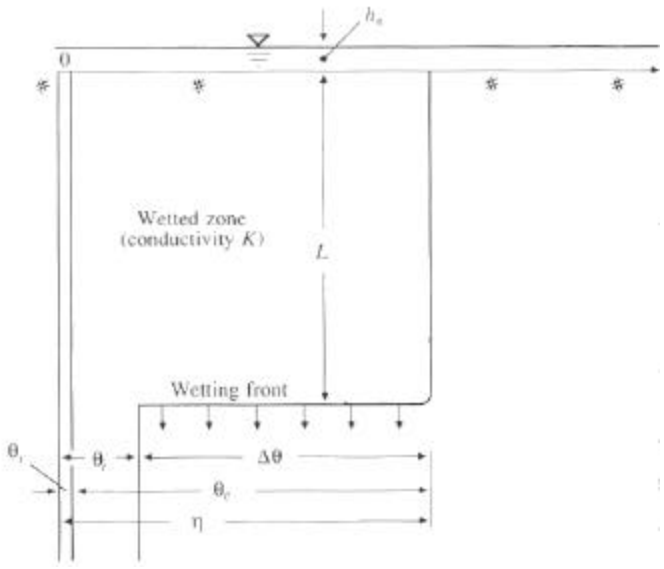


Figure 1: Simple model representing infiltration of surface water into the soil horizon.

The Green & Ampt model assumes that infiltration occurs from a ponded surface of depth h_0 into a deep soil column of length L. The water enters the soil as a slug according to Darcy's Law ($f = K\,\partial h/\partial z$). Furthermore, the model assumes a sharp wetting front separating a wetted zone from an unwetted zone with an initial moisture content θ_i. According to these model assumptions, infiltration can calculated from the expression:

$$f(t) = K\left(\frac{\Psi \Delta q}{F(t)} + 1\right)$$

where K is the hydraulic conductivity, $\Delta\theta$ is the change in moisture content, Ψ is the wetting front suction head, and F(t) is the cumulative depth of water infiltrated.

With the use of the State Soil Geographic Database (STATSGO) it is possible to assign Green and Ampt parameters to a watershed of interest. Furthermore, this database allows for a general soil analysis of a given area including percentages of the various soil textures.

Chapter 2 Methodology

2.1 Study Area

The study area used is the Guadalupe basin located in central Texas. This area was selected because the data necessary for delineation of this basin was readily available.

2.2 Required Input Data

The data required to analyze the Guadalupe basin included:

- Delineated Watershed Theme
- A user input effective saturation (S_e) for the entire watershed.
- A STATSGO map of soil textures for the study area supplied by the National STATSGO database.
- Database files called *mapunit.dbf*, *comp.dbf* and *layer.dbf* supplied by the National STATSGO database. The *mapunit* table holds the key to relating soil attributes to polygons on the map. One record exists in the *mapunit* table for each *mapunit*. A single record in the *mapunit* table relates to one or more components in the *comp* table and the *layer* table. For example (see figure 3), the mapunit TX001 is made

up of four (4) components named Acadia, Aldine, Waller and Harris. Furthermore, each of these components are made up of layers, Acadia has four (4) layers, Aldine has three (3) layers etc. (Both the soils map and the tables are available through the National STATSGO database at http://www.ftw.nrcs.usda.gov/stat_data.html

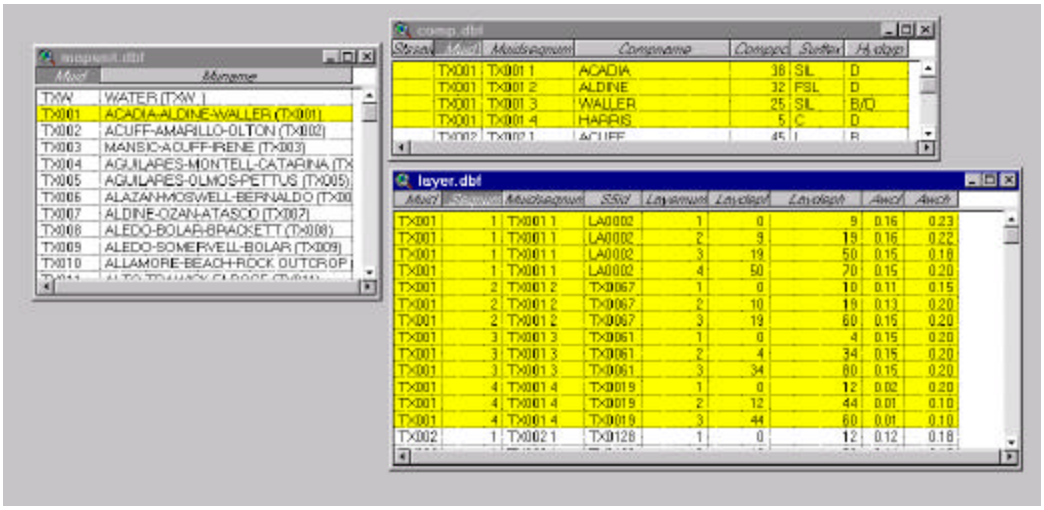


Figure 2: Database files necessary for a soil analysis.

With the use of CRWRprepro the Guadalupe basin was delineated and the STATSGO soil texture map was clipped to include only the study area. (To see the steps necessary to delineate a watershed and clip the necessary soil texture data refer to http://www.engr.utexas.edu/giswr/secure/Module3/exercise3/delinex.htm and http://www.engr.utexas.edu/giswr/secure/Module5/exercise5/landsoil.htm). The delineated watershed and the clipped soil texture data can be seen below in Figure 3.

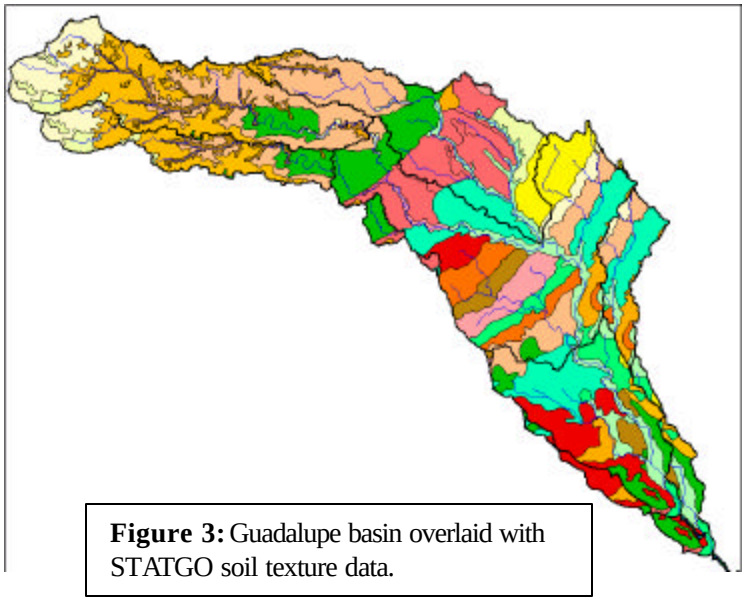


Figure 3: Guadalupe basin overlaid with STATGO soil texture data.

2.3 Creating an ArcView GIS Script using Avenue

To assign Green and Ampt parameters to the watershed of interest, an Avenue script was created. This script first links the three STATSGO tables. By linking the tables it is possible to pull required data from thousands of records to produce a table called *USDA.dbf* (see Figure 4). This table acts as a modified component table and is the source of all further calculations.

USDA.dbf

Muid	Muidsegnum	Compname	Surftex	Hydgrp	Comppct	USDA	Cond	EffPor	SuctHead	HoldCap
TXW	TXW 1	WATER			100		0.000	0.000	0.000	0.000
TX001	TX001 1	ACADIA	SIL	D	38	SIL	6.500	0.486	166.800	311.658
TX001	TX001 2	ALDINE	FSL	D	32	SL	10.900	0.412	110.100	252.984
TX001	TX001 3	WALLER	SIL	B/D	25	SIL	6.500	0.486	166.800	355.600
TX001	TX001 4	HARRIS	C	D	5	C	0.300	0.385	316.300	100.584
TX002	TX002 1	ACUFF	L	B	45	L	3.400	0.434	88.900	293.370
TX002	TX002 2	ACUFF	L	B	5	L	3.400	0.434	88.900	293.370
TX002	TX002 3	AMARILLO	FSL	B	20	SL	10.900	0.412	110.100	279.400
TX002	TX002 4	AMARILLO	FSL	B	15	SL	10.900	0.412	110.100	279.400
TX002	TX002 5	OLTON	CL	C*	10	CL	1.000	0.309	208.800	319.024
TX002	TX002 6	RANDALL	C	D	5	C	0.300	0.385	316.300	247.396
TX003	TX003 1	ACUFF	L	B	12	L	3.400	0.434	88.900	293.370
TX003	TX003 2	ACUFF	L	B	5	L	3.400	0.434	88.900	293.370
TX003	TX003 3	AMARILLO	FSL	B	8	SL	10.900	0.412	110.100	279.400
TX003	TX003 4	AMARILLO	FSL	B	1	SL	10.900	0.412	110.100	279.400
TX003	TX003 5	OLTON	CL	C*	5	CL	1.000	0.309	208.800	308.864

Figure 4: Modified component Table known as "*USDA.dbf*"

The *USDA.dbf* table is created by converting the 719 USGS surface texture codes (surftex) to the 12 USDA codes (See Figure 5). This is done from a texture conversion table (See Figure 6). If there is not a direct conversion, the hydrologic group (hydgrp) is used (A, B, C and D) to estimate parameters.

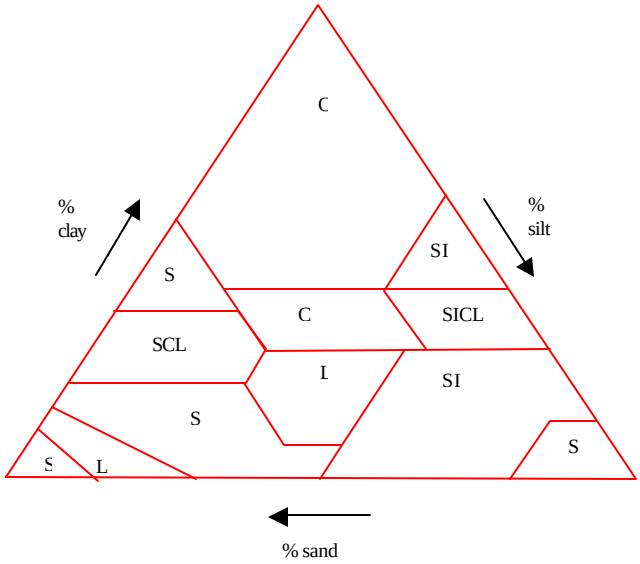


Figure 5: USDA soil texture triangle.

textconv.dbf

Statsgo_na	Usda_textu
BY-C	C
BY-CL	CL
BY-COS	S
BY-COSL	SL
BY-FS	S
BY-FSL	SL
BY-L	L
BY-LCOS	LS
BY-LS	LS
BY-LVFS	LS
BY-S	S
BY-SC	SC
BY-SCL	SCL

Figure 6: Table used to convert from USGS soil textures to USDA soil textures.

Once the USDA soil textures are known for each component, a look-up table (See Figure 7) is used to assign Green and Ampt parameters to each component in the table.

lookuptable.dbf

Soil_class	Usda_code	Effective_	Wetting_fr	Hydraulic_
Sand	S	0.417	49.5	117.8
Loamy Sand	LS	0.401	61.3	29.9
Sandy Loam	SL	0.412	110.1	10.9
Loam	L	0.434	88.9	3.4
Silt Loam	SIL	0.486	166.8	6.5
Sandy Clay Lo	SCL	0.33	218.5	1.5
Clay Loam	CL	0.309	208.8	1.0
Silty Clay Lo	SICL	0.432	273.0	1.0
Sandy Clay	SC	0.321	239.0	0.6
Silty Clay	SIC	0.423	292.2	0.5
Clay	C	0.385	316.3	0.3
Silt	SI	0.486	166.8	6.5
Freely Draini	A	0.417	49.5	117.8
Intermediate	B	0.434	88.9	3.4
Intermediate	C*	0.309	208.8	1.0
Poorly Draini	D	0.385	316.3	0.3

Figure 7: Look-up table of Green and Ampt parameters for the 12 USDA soil textures

The parameters in the look-up table include the effective porosity (θ_e), the wetting front suction head (ψ) in mm and the hydraulic conductivity (K) in mm/hr. The final value in the modified component table is the holding capacity.

This value is calculated from the layer table with the expression:

$$\text{Holding Capacity} = \frac{(\text{awcl} + \text{awch})}{2} * (\text{laydeph} - \text{laydepl})$$

where laydepl is the depth of the top of the layer in inches, laydeph is the depth of the bottom of the layer in inches, awcl is the lower limit on the estimated water holding capacity in inches of water per inch depth of soil (e.g. a value of 0.16 in/in means that 16 per cent of the soil volume is void space that could be occupied by water), and awch is the upper limit on the estimate water holding capacity. The values in the layer table are in inches and converted by the avenue script to millimeters.

The modified component table (*USDA.dbf*) contains parameters for each component, while each mapunit is composed of many components. This data must be further processed to give only one set of parameters for each mapunit found in a watershed. The parameters for each mapunit are estimated by using an area-weighted percentage. These percentages can be found in the modified component table under the field "*comppct*". The expressions used to calculate the parameters for each mapunit are as follows:

$$\text{Conductivity} = \sum_{\text{component}} \text{Conductivity} * \frac{\text{comppct}}{100}$$

$$\text{Suction Head} = \sum_{\text{component}} \text{Suction Head} * \frac{\text{comppct}}{100}$$

$$\text{Change in Moisture Content } (\Delta q) = \sum_{\text{component}} (1 - S_e) * \text{Effective Porosity} * \frac{\text{comppct}}{100}$$

$$\text{Initial Loss} = \sum_{\text{component}} (1 - S_e) * \text{Holding Capacity} * \frac{\text{comppct}}{100} * 0.2$$

where S_e the user input effective saturation of the watershed.

These weighted parameters are assigned to the mapunits of the soils attribute table (See Figure 8). Also included in the soils attribute table, is the percentage of each of the soil types and the percentage of water located within each mapunit (See Figure 9).

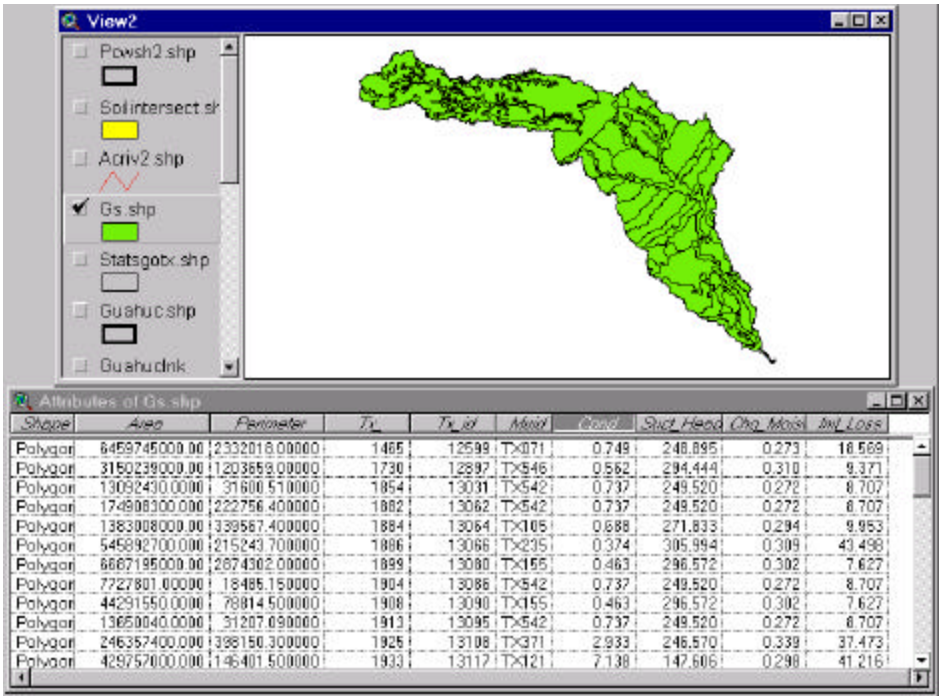
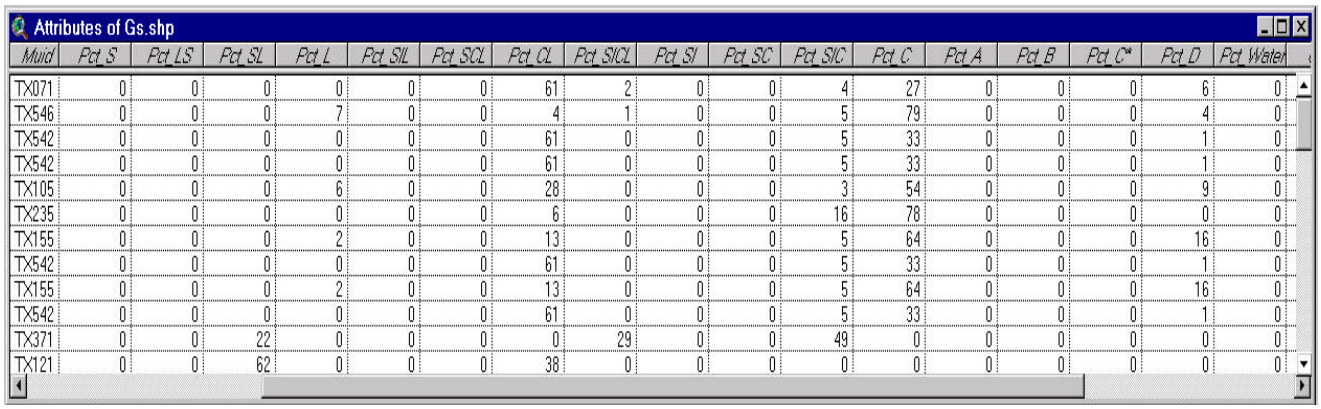


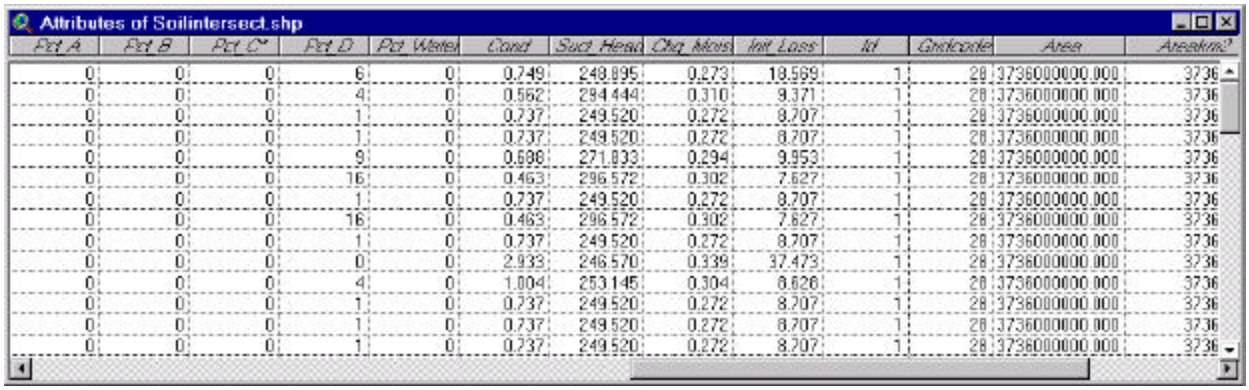
Figure 8: Area weighted Green and Ampt parameters assigned to each mapunit in an area of interest.



Muid	Pct_S	Pct_LS	Pct_SL	Pct_L	Pct_SIL	Pct_SCL	Pct_CL	Pct_SICL	Pct_SI	Pct_SC	Pct_SIC	Pct_C	Pct_A	Pct_B	Pct_C*	Pct_D	Pct_Water
TX071	0	0	0	0	0	0	61	2	0	0	4	27	0	0	0	6	0
TX546	0	0	0	7	0	0	4	1	0	0	5	79	0	0	0	4	0
TX542	0	0	0	0	0	0	61	0	0	0	5	33	0	0	0	1	0
TX542	0	0	0	0	0	0	61	0	0	0	5	33	0	0	0	1	0
TX105	0	0	0	6	0	0	28	0	0	0	3	54	0	0	0	9	0
TX235	0	0	0	0	0	0	6	0	0	0	16	78	0	0	0	0	0
TX155	0	0	0	2	0	0	13	0	0	0	5	64	0	0	0	16	0
TX542	0	0	0	0	0	0	61	0	0	0	5	33	0	0	0	1	0
TX155	0	0	0	2	0	0	13	0	0	0	5	64	0	0	0	16	0
TX542	0	0	0	0	0	0	61	0	0	0	5	33	0	0	0	1	0
TX371	0	0	22	0	0	0	0	29	0	0	49	0	0	0	0	0	0
TX121	0	0	62	0	0	0	38	0	0	0	0	0	0	0	0	0	0

Figure 9: Analysis of the soils for each mapunit in a watershed of interest. The table includes the percentages of each of the twelve (12) USDA soil types, percentages of the hydrologic groups for unrecognized soil types and the percentage of water.

After the Green & Ampt parameters have been assigned to the soils attribute table, the soils theme and the watershed theme are intersected. This intersection splits mapunits that cross a watershed boundary and assigns a subwatershed code known as a gridcode to each polygon. The intersected theme is then put into the active view and the new attribute table includes the recalculated areas of each polygon and its corresponding gridcode (See Figure 10).


Attributes of Soilintersect.shp

Pct_A	Pct_B	Pct_C*	Pct_D	Pct_Water	Cond	Suct_Head	Chg_Mois	Init_Loss	Id	Gridcode	Area	Areakm2
0	0	0	6	0	0.749	248.895	0.273	18.569	1	28	3736000000.000	3736
0	0	0	4	0	0.562	284.444	0.310	9.371	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736
0	0	0	9	0	0.696	271.033	0.294	9.953	1	28	3736000000.000	3736
0	0	0	16	0	0.463	296.572	0.302	7.627	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736
0	0	0	16	0	0.463	296.572	0.302	7.627	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736
0	0	0	0	0	2.933	246.570	0.339	37.473	1	28	3736000000.000	3736
0	0	0	4	0	1.004	253.145	0.304	8.626	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736
0	0	0	1	0	0.737	249.520	0.272	8.707	1	28	3736000000.000	3736

Figure 10: Attribute table of the new theme created from intersecting the watershed theme and the soils theme.

With the use of the gridcode and the recalculated areas it is possible to estimate area weighted Green & Ampt parameters for each subwatershed in the study area. Each parameter is estimated using an expression similar to the equation below.

$$\text{Conductivity} = \sum_{\text{gridcode}} \text{Conductivity} * \frac{\text{mapunit area}}{\text{subwatershed area}}$$

Finally, the weighted Green & Ampt parameters are entered into the watershed attribute table (See Figure 11). This table is the final output of the Avenue script.

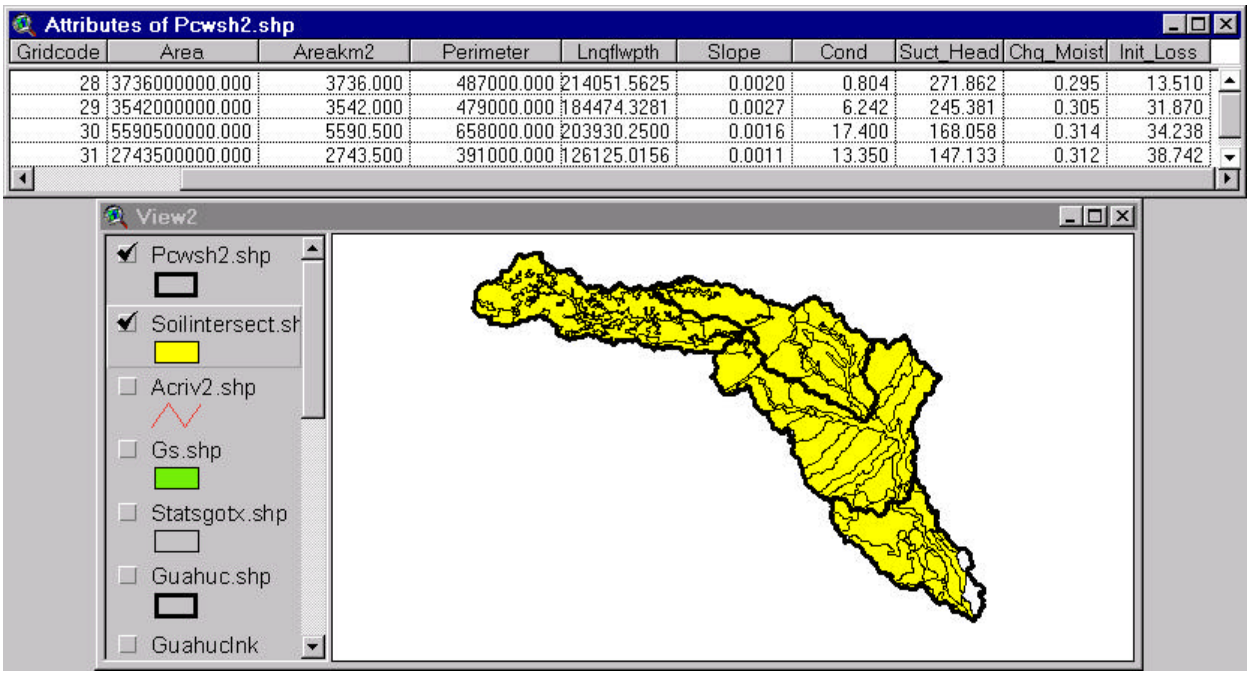


Figure 11: New watershed attribute table that includes Green and Ampt parameters

Chapter 3 Conclusions and Future Work

The Avenue script that has been created automates the process necessary to assign Green & Ampt parameters to a delineated watershed. Furthermore, the script allows the user to easily analyze the soil textures that can be found within a particular watershed.

With Green and Ampt parameters assigned to each subwatershed, the possibility of creating a basin file for use in HEC-HMS that includes these parameters is the next step. This step can be accomplished by incorporating this script into CRWRprepro.

Appendix A: Processing a Delineated Watershed to include Green and Ampt Infiltration Parameters

The intent of this exercise is to introduce you to using the STATSGO soil texture maps and the related data tables to produce a delineated watershed that includes Green and Ampt infiltration parameters in the watershed's attribute table.

Obtaining the Data

The data required include:

- ***Statsgotx.shp*** - a shape file of Statsgo map units for Texas

- Database files called ***mapunit.dbf***, ***comp.dbf*** and ***layer.dbf*** supplied by the National STATSGO database. The *mapunit* table holds the key to relating soil attributes to polygons on the map. One record exists in the *mapunit* table for each *mapunit*. A single record in the *mapunit* table relates to one or more components in the *comp* table and the *layer* table. The data tables are available through the National STATSGO database at ftp://ftp.ftw.nrcs.usda.gov/pub/statsgo/dos/arc/data/.

- ***Green.apr*** - an ArcView project file that contains the Script for processing the delineated watershed

- ***Watershed.shp*** - delineated watershed (the Buffalo Bayou basin)

- ***Textconv.dbf*** - a table used to convert the 719 USGS soil textures to the 12 USDA textures

- ***Lookuptable.dbf*** - a table used to assign Green and Ampt infiltration parameters based on the 12 USDA textures

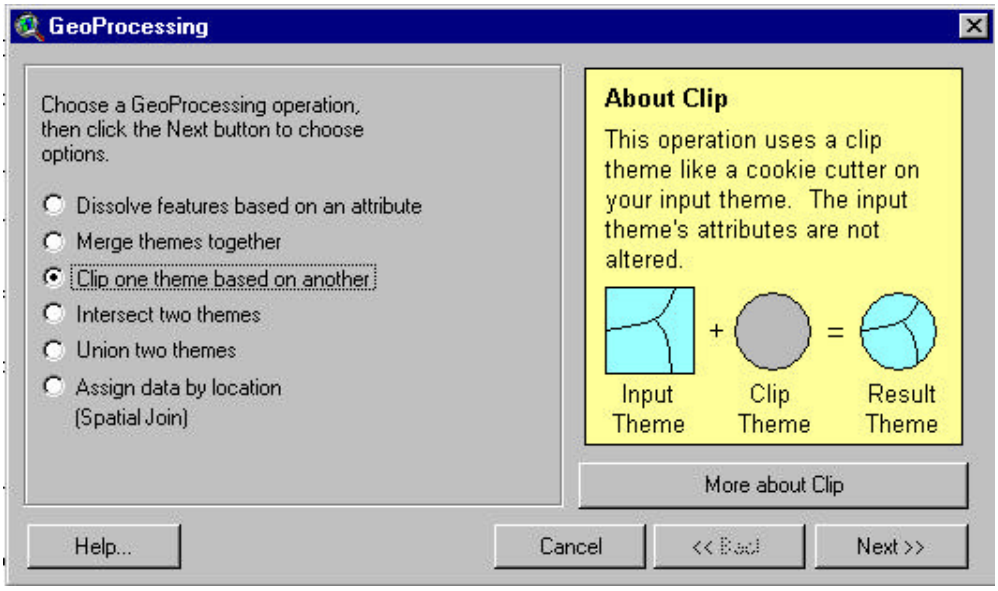
All of these files can be downloaded as Green.zip (8.7 MB). This file should be unzipped in your working directory.

Procedure

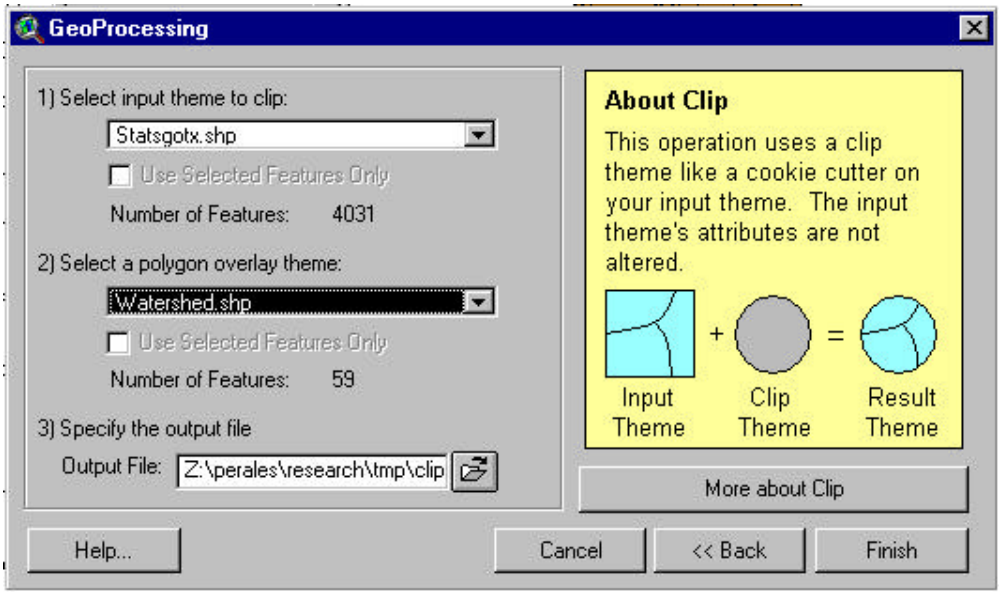
1. Clipping the Soils Data to the Watershed of Interest

Open the ***Green.apr*** file and name your working directory. Your files from ***Green.zip*** should be located here. Open a new view and add the themes ***Statsgotx.shp*** and ***Watershed.shp***. Next add the tables ***mapunit.dbf***, ***comp.dbf***, ***layer.dbf***, ***textconv.dbf*** and ***lookuptable.dbf***.

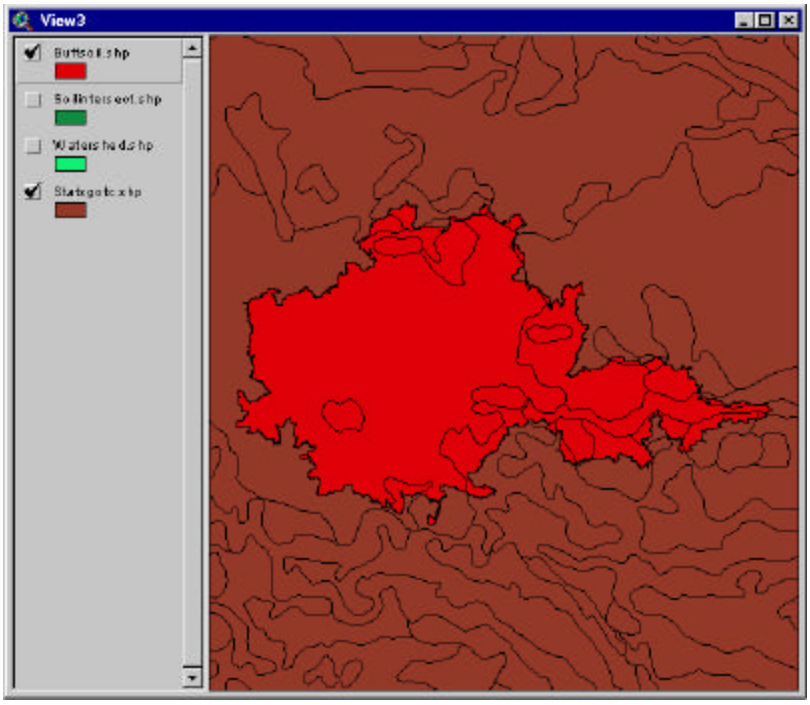
To use the clipping tool the geoprocessing extension must be loaded. This can be done from the **File** menu under **Extensions.** Once the Extensions dialog box has opened, scroll down and select geoprocessing. Then with the view the active window, under the **View** menu select **GeoProcessing Wizard**. With the GeoProcessing Wizard open, select *clip one theme based on another* and push the next button.



The input theme is the Statsgotx.shp file and the overlay theme is the watershed of interest.

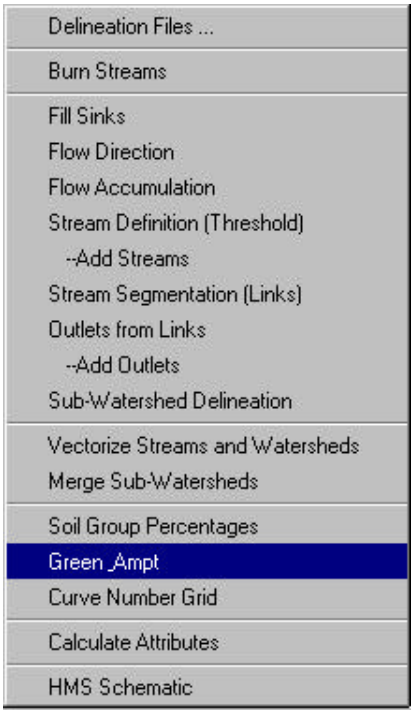


Then press the finish button and a clipped theme will be put into your view. You should give the clipped soils theme a name you will recognize.



2. Using the Green and Ampt Script

To use the script go to the **CRWR-PrePro** menu and select **Green_Ampt**.



There will be a series of dialog boxes that require input themes and tables. Follow the instructions on the dialog boxes. The final dialog box will ask for the *Effective Saturation* of the watershed. The default value is 20% but can be changed.

The program will run for a few minutes. When it is finished a table called ***USDA.dbf*** will have been added to the list of tables and then you will be prompted to add a new theme to the view. This new theme that is to be added will automatically be called ***Soilintersect.shp.*** Open the attribute table for the ***soilintersect*** shapefile. To do this make sure the appropriate theme is active and then push the open theme table button  . Notice that for each mapunit the soils are divided into thirteen (13) categories, the twelve (12) USDA soil types and water. Furthermore, the Green and Ampt parameters of hydraulic conductivity, suction head, change in moisture content, and initial loss have

been added to the table. All of these values have been reduced from the ***USDA.dbf*** table.

Open the ***USDA.dbf*** table. In this table you will notice that each mapunit is made-up of different components. Moreover, each component is made-up of layers. This table is used to assign Green and Ampt parameters to each component. This table acts as a modified component table and is the source of all further calculations. The values for conductivity, suction head and effective porosity are pulled directly from the look-up table. The value of the holding capacity is calculated from the layers table from the expression:

Holding Capacity = sum over its layers of (awcl + awch)/2 * (laydeph - laydepl).

where:

- **Laydepl**- the depth of the top of the layer in inches.
- **Laydeph**- the depth of the bottom of the layer in inches.
- **Awcl**- a lower limit on the estimated water holding capacity in inches of water per inch depth of soil (e.g. a value of 0.16 in/in means that 16 per cent of the soil volume is void space that could be occupied by water).
- **Awch** - an upper limit on the estimate water holding capacity.

These parameters are further reduced in the soils attribute table by using an area weighted average. In the ***USDA.dbf*** table the parameter *comppct* represents an area weighted percentage of each component that makes- up a mapunit. The area weighted hydraulic conductivity and the wetting front suction head values are calculated from the expressions below:

$$\text{Conductivity} = \sum_{\text{component}} \text{Conductivity} * \frac{\text{comppct}}{100}$$

$$\text{Suction Head} = \sum_{\text{component}} \text{Suction Head} * \frac{\text{comppct}}{100}$$

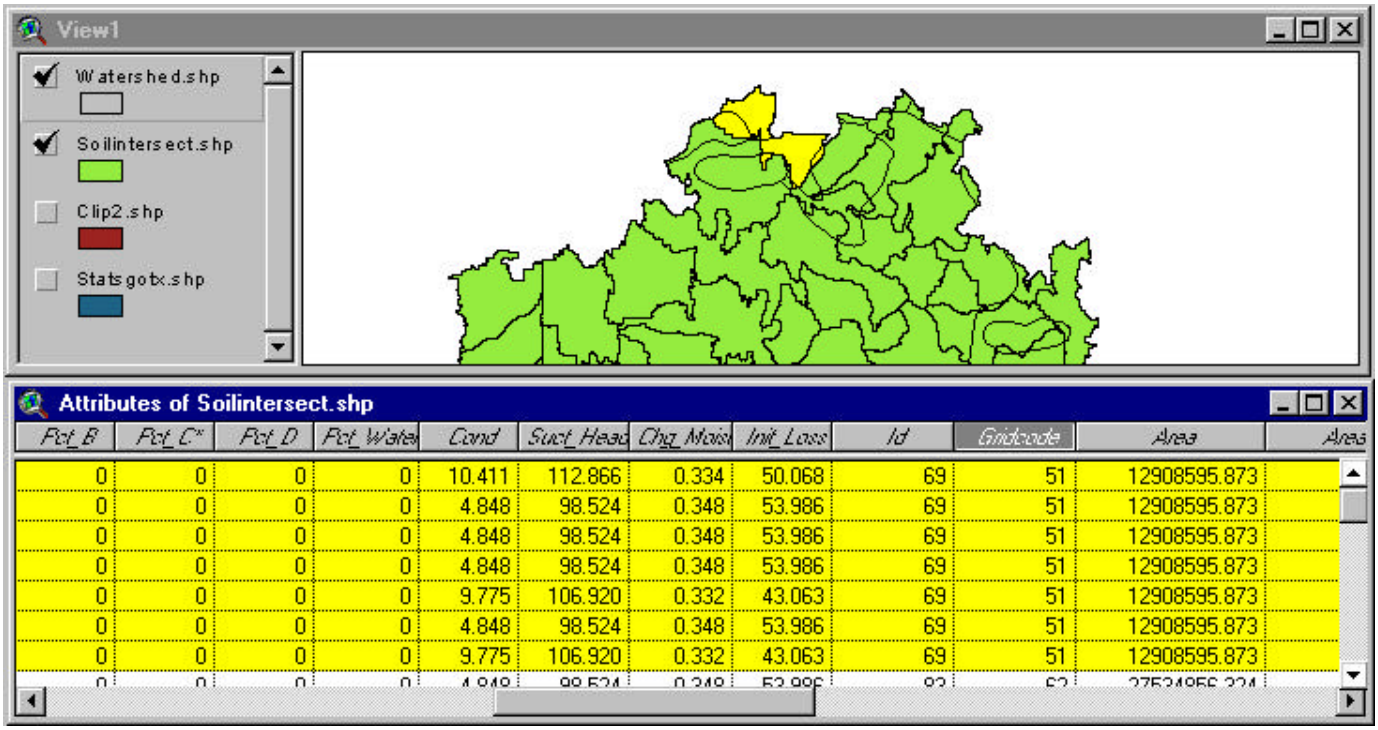
The Change in Moisture Content ($\Delta\theta$) and the Initial Loss values are estimated from a user input Effective Saturation (S_e) value and are area weighted.

$$\Delta\theta = \sum_{\text{component}} (1 - S_e) * \text{Effective Porosity} * \frac{\text{comppct}}{100}$$

$$\text{Initial Loss} = \sum_{\text{component}} (1 - S_e) * \text{Holding Capacity} * \frac{\text{comppct}}{100} * 0.2$$

These weighted parameters are assigned to the mapunits of the ***soil attributes table***. Scroll through the clipped soils attribute table and notice there is only one value for each mapunit. Also included in the ***soil attributes table***, is the percentage of each of the soil types and the percentage of water located within each mapunit.

Now, with the ***soilintersect attributes table*** in view, select the field labeled gridcode and press the sort ascending button  . Hold the shift button down and select all rows with the same gridcode. In the view the subwatershed with this gridcode will be



highlighted in yellow. Notice that the ***soilintersect attributes table*** contains both mapunits and gridcodes. This table provides the connection needed to assign soil parameters to a watershed.

Notice that this table has two area values. The first area value is of the individual polygon of the mapunit and the second area is the area of the entire subwatershed that the mapunit is a part of. The first area value is used to further reduce the data so that each gridcode has only one set of parameters. This set of parameters are then entered into the watershed attributes table. Open the watershed attribute table and select the gridcode field and press the sort ascending button. Notice that there is only one value for each gridcode. Again, this reduction was done by taking the sum of the area weighted mapunit values.

Appendix B: Avenue Script for Green and Ampt Soil Analysis

```
'=========================================================
' Name: txdot.g&aparam'
' Description: Processes STATSGO data to include Green
and Ampt parameters
' in the watershed attribute table
'
' Requires: STATSGO Soils Map with supporting mapunit,
component and layer tables.
'           Also needed is a delineated watershed
' History: Created by Jerome Perales
'=========================================================
===
theProject=av.getproject
'---------------------------------------------------------
-
'---IDENTIFY INPUTs
'---------------------------------------------------------
-
theview=av.getactivedoc
thedir=theproject.getworkdir
thethemes=theview.getthemes
vectthemes=list.make
for each t in thethemes
   if (t.getclass.getclassname <> "GTheme") then
      vectthemes.add(t)
   end
end

'---choose the soils theme---

soilTheme=Msgbox.Choiceasstring(vectthemes,"Choose the
soil Theme.","Soil Group")
if (soilTheme = NIL) then exit end
soilftab=soiltheme.getftab
soilftab.seteditable(true)

'---test the soils theme for the mapunit id field---

muidsf=soilftab.findfield("muid")
if (muidsf=nil) then
  msgbox.error("Could not find map unit id.","Soil
Group")
  exit
end

'----------------------------------
'--- add  fields to soils table ---
'----------------------------------
```

```
addflist1=list.make

fPct_S=soilftab.findfield("Pct_S")
if (fPct_S = nil) then
addflist1.add(field.make("Pct_S", #field_decimal,8,0))
end


fPct_LS=soilftab.findfield("Pct_LS")
if (fPct_LS = nil) then
addflist1.add(field.make("Pct_LS", #field_decimal,8,0))
end


fPct_SL=soilftab.findfield("Pct_SL")
if (fPct_SL = nil) then
addflist1.add(field.make("Pct_SL", #field_decimal,8,0))
end


fPct_L=soilftab.findfield("Pct_L")
if (fPct_L = nil) then
addflist1.add(field.make("Pct_L", #field_decimal,8,0))
end


fPct_SIL=soilftab.findfield("Pct_SIL")
if (fPct_SIL = nil) then
addflist1.add(field.make("Pct_SIL", #field_decimal,8,0))
end


fPct_SCL=soilftab.findfield("Pct_SCL")
if (fPct_SCL = nil) then
addflist1.add(field.make("Pct_SCL", #field_decimal,8,0))
end


fPct_CL=soilftab.findfield("Pct_CL")
if (fPct_CL = nil) then
addflist1.add(field.make("Pct_CL", #field_decimal,8,0))
end


fPct_SICL=soilftab.findfield("Pct_SICL")
if (fPct_SICL = nil) then
addflist1.add(field.make("Pct_SICL", #field_decimal,8,0))
end
```

```
fPct_SI=soilftab.findfield("Pct_SI")
if (fPct_SI = nil) then
addflist1.add(field.make("Pct_SI", #field_decimal,8,0))
end


fPct_SC=soilftab.findfield("Pct_SC")
if (fPct_SC = nil) then
addflist1.add(field.make("Pct_SC", #field_decimal,8,0))
end


fPct_SIC=soilftab.findfield("Pct_SIC")
if (fPct_SIC = nil) then
addflist1.add(field.make("Pct_SIC", #field_decimal,8,0))
end


fPct_Clay=soilftab.findfield("Pct_C")
if (fPct_Clay = nil) then
addflist1.add(field.make("Pct_C", #field_decimal,8,0))
end


fPct_A=soilftab.findfield("Pct_A")
if (fPct_A = nil) then
addflist1.add(field.make("Pct_A", #field_decimal,8,0))
end


fPct_B=soilftab.findfield("Pct_B")
if (fPct_B = nil) then
addflist1.add(field.make("Pct_B", #field_decimal,8,0))
end


fPct_C=soilftab.findfield("Pct_C*")
if (fPct_C = nil) then
addflist1.add(field.make("Pct_C*", #field_decimal,8,0))
end


fPct_D=soilftab.findfield("Pct_D")
if (fPct_D = nil) then
addflist1.add(field.make("Pct_D", #field_decimal,8,0))
end


fPct_W=soilftab.findfield("Pct_Water")
if (fPct_W = nil) then
```

```
addflist1.add(field.make("Pct_Water",
#field_decimal,8,0))
end


scondf=soilftab.findfield("Cond")
if (scondf = nil) then
addflist1.add(field.make("Cond", #field_decimal,8,3))
end


ssuctf=soilftab.findfield("Suct_Head")
if (ssuctf = nil) then
addflist1.add(field.make("Suct_Head",
#field_decimal,8,3))
end


schgmoistf=soilftab.findfield("Chg_Moist")
if (schgmoistf = nil) then
addflist1.add(field.make("Chg_Moist",
#field_decimal,8,3))
end


sinitlossf=soilftab.findfield("Init_Loss")
if (sinitlossf = nil) then
addflist1.add(field.make("Init_Loss",
#field_decimal,8,3))
end




if ((addflist1.count = 0).not) then
  soilftab.addfields(addflist1)
end
addflist1.empty



'---choose watershed theme---


watershedTheme=Msgbox.Choiceasstring(vectthemes,"Choose
the Watershed Theme.","Soil Group")
if (watershedTheme = NIL) then exit end
wshftab=watershedTheme.getftab
wshftab.seteditable(true)

'---test watershed theme for gridcode id---
```

```
gridcodef=wshftab.FindField("Gridcode")
if (gridcodef=nil) then
  gridcodef=wshftab.FindField("Grid_code")
end

'---identify the STATSTGO tables mapunit.dbf comp.dbf and
layer.dbf---

tablelist=list.make
for each d in theproject.getdocs
  if (d.getgui="Table") then
     tablelist.add(d)
  end
end

foundmuid=false
foundcomp=false
foundlayer=false



for each t in tablelist

if (t.asstring = "mapunit.dbf") then
  muidtab="mapunit.dbf"
  foundmuid=true
  muidvtab=t.getvtab
end

if (t.asstring = "comp.dbf") then
  comptab="comp.dbf"
  foundcomp=true
  compvtab=t.getvtab
end

if (t.asstring = "layer.dbf") then
  layertab="layer.dbf"
  foundlayer=true
  layervtab=t.getvtab
end

end

if (not foundmuid) then
  muidtab=msgbox.choiceasstring(tablelist,"Identify the
Mapunit Table.","Soil Group")

  if (muidtab=nil) then
    msgbox.error("Could not find map unit table.","Soil
Group")
```

```
    exit
  end
  layervtab=layertab.getvtab
end


if (not foundcomp) then
  comptab=msgbox.choiceasstring(tablelist,"Identify the
Component Table.","Soil Group")

  if (comptab=nil) then
    msgbox.error("Could not find component table.","Soil
Group")
    exit
  end
  compvtab=comptab.getvtab
end

if (not foundlayer) then
  layertab=msgbox.choiceasstring(tablelist,"Identify the
Layer Table.","Soil Group")

  if (layertab=nil) then
      msgbox.error("Could not find layer table.","Soil
Group")
      exit
  end
  layervtab=layertab.getvtab
end

'---choose texture conversion table---

txtconvtab=msgbox.choiceasstring(tablelist,"Identify the
Soil Texture Conversion Table.","Soil Group")
if (txtconvtab=nil) then
  msgbox.error("Could not find soil texture conversion
table.","Soil Group")
  exit
end

txtcon=txtconvtab.getvtab
if (txtcon=nil) then
   msgbox.error("Error selecting Texture Conversion
Vtab.","")
   exit
 end


'---choose lookup table---
```

```
lookuptab=msgbox.choiceasstring(tablelist,"Identify the
Look-Up Table.","Soil Group")
if (lookuptab=nil) then
  msgbox.error("Could not find look-up table.","Soil
Group")
  exit
end

lookupvtab=lookuptab.getvtab
if (lookupvtab=nil) then
   msgbox.error("Error selecting Lookup Vtab.","")
   exit
 end


'---find selected fields---

statsgo=txtcon.findfield("statsgo_na")
USDA=txtcon.findfield("USDA_textu")
usda_code=lookupvtab.findfield("Usda_code")

'--- clear all previous selections and links---

if (muidvtab=nil) then
   msgbox.error("Error selecting mapunit Vtab.","")
   exit
 end


 if (layervtab=nil) then
   msgbox.error("Error selecting component Vtab.","")
   exit
 end


 if (compvtab=nil) then
   msgbox.error("Error selecting component Vtab.","")
   exit
 end


 muidbit=muidvtab.getselection
 compbit=compvtab.getselection
 layerbit=layervtab.getselection

 muidbit.clearall
 compbit.clearall
 layerbit.clearall

 muidvtab.updateselection
 compvtab.updateselection
```

```
 layervtab.updateselection

 if (muidvtab.islinked) then
    muidvtab.unlinkall
 end

 if (compvtab.islinked) then
   compvtab.unlinkall
 end

if (layervtab.islinked) then
   layervtab.unlinkall
 end
'-------------------------------------------------------
-
'--- create a table containing mapunit IDs, Surftex,
hydrogroup data
'--- and Green & Ampt variables
'-------------------------------------------------------
-
odoc=theproject.finddoc("USDA.dbf")
if (odoc<>nil) then
  theproject.removedoc(odoc)
end
addflist=list.make
addflist.add(muidvtab.findfield("muid").clone)
addflist.add(compvtab.findfield("muidseqnum").clone)
addflist.add(compvtab.findfield("compname").clone)
addflist.add(compvtab.findfield("surftex").clone)
addflist.add(compvtab.findfield("hydgrp").clone)
addflist.add(compvtab.findfield("comppct").clone)
addflist.add(field.make("USDA", #field_char,8,0))
addflist.add(field.make("Cond", #field_decimal,8,3))
addflist.add(field.make("EffPor", #field_decimal,8,3))
addflist.add(field.make("SuctHead", #field_decimal,8,3))
addflist.add(field.make("HoldCap", #field_decimal,8,3))
joinfilename=fn.merge(thedir.asstring,"USDA.dbf")

gavtab=vtab.makenew(joinfilename,dBase)
gavtab.seteditable(true)
gavtab.addfields(addflist)
outfields=gavtab.getfields
outtab=table.make(gavtab)
outtab.setname("USDA.dbf")

'-------------------------------------------------------
-
'--- identify input fields needed for computations
'-------------------------------------------------------
-
fPct_S=soilftab.findfield("Pct_S")
```

```
fPct_LS=soilftab.findfield("Pct_LS")
fPct_SL=soilftab.findfield("Pct_SL")
fPct_L=soilftab.findfield("Pct_L")
fPct_SIL=soilftab.findfield("Pct_SIL")
fPct_SCL=soilftab.findfield("Pct_SCL")
fPct_CL=soilftab.findfield("Pct_CL")
fPct_SICL=soilftab.findfield("Pct_SICL")
fPct_SI=soilftab.findfield("Pct_SI")
fPct_SC=soilftab.findfield("Pct_SC")
fPct_SIC=soilftab.findfield("Pct_SIC")
fPct_Clay=soilftab.findfield("Pct_C")
fPct_A=soilftab.findfield("Pct_A")
fPct_B=soilftab.findfield("Pct_B")
fPct_C=soilftab.findfield("Pct_C*")
fPct_D=soilftab.findfield("Pct_D")
fPct_W=soilftab.findfield("Pct_Water")
scondf=soilftab.findfield("Cond")
ssuctf=soilftab.findfield("Suct_Head")
schgmoistf=soilftab.findfield("Chg_Moist")
sinitlossf=soilftab.findfield("Init_Loss")

condf=lookupvtab.findfield("hydraulic_")
if (condf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end


suctf=lookupvtab.findfield("wetting_fr")
if (suctf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

effporf=lookupvtab.findfield("effective_")
if (effporf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

muidf=muidvtab.findfield("muid")
if (muidf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

compnamef=compvtab.findfield("compname")
if (compnamef=nil) then
  msgbox.info("Cannot find a field","")
  exit
end
```

```
surftexf=compvtab.findfield("surftex")
if (surftexf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

hydgrpf=compvtab.findfield("hydgrp")
if (hydgrpf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

comppctf=compvtab.findfield("comppct")
if (comppctf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

muidseqnumf=compvtab.findfield("Muidseqnum")
if (muidseqnumf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

laydeplf=layervtab.findfield("Laydepl")
if (laydeplf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

laydephf=layervtab.findfield("Laydeph")
if (laydephf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

awclf=layervtab.findfield("Awcl")
if (awclf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

awchf=layervtab.findfield("Awch")
if (awchf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

lmuidseqnumf=layervtab.findfield("muidseqnum")
if (lmuidseqnumf=nil) then
```

```
  msgbox.info("Cannot find a field","")
  exit
end

holdcapf=gavtab.findfield("holdcap")
if (lmuidseqnumf=nil) then
  msgbox.info("Cannot find a field","")
  exit
end

gcondf=gavtab.findfield("Cond")
if (gcondf=nil) then
  msgbox.info("Cannot find Hydraulic Conductivity field
in the Output table","")
  exit
end

geffporf=gavtab.findfield("EffPor")
if (geffporf=nil) then
  msgbox.info("Cannot find Effective Porosity field in
the Output table","")
  exit
end

gsuctf=gavtab.findfield("SuctHead")
if (gsuctf=nil) then
  msgbox.info("Cannot find Suction Head field in the
Output table","")
  exit
end

convdict = dictionary.make(2*(txtcon.getnumrecords))
for each r in txtcon

      akey=txtcon.returnvalue(statsgo,r)
      avalue=txtcon.returnvalue(USDA,r)
      convdict.set(akey, avalue)

   end

s = msgbox.input("Please input the Effective Saturation
(0 - 1)","Effective Saturation","0.2").AsNumber

templist=list.make
complist=list.make

'-----------------------------------------------------
'--- link the mapunit table to the component table---
'-----------------------------------------------------
'--- identify link fields
```

```
tofield1=compvtab.findfield("muid")
if (tofield1=nil) then
   msgbox.info("Could not find field muid in component
table.","")
   exit
end

tofield2=layervtab.findfield("muid")
if (tofield2=nil) then
   msgbox.info("Could not find field muid in layer
table.","")
   exit
end



fromfield1=muidvtab.findfield("muid")
if (fromfield1=nil) then
   msgbox.info("Could not find field muid in the mapunit
table.","")
   exit
end
muidvtab.link(fromfield1,compvtab,tofield1)
muidvtab.link(fromfield1,layervtab,tofield2)


'-------------------------------------------------------
-----
'--- populate the USDA.dbf table from the data in
comp.dbf ---
'--- layer.dbf and mapunit.dbf--------------------------
-----
'-------------------------------------------------------
-----
numrecs=muidvtab.getnumrecords
muidcount=0
'--- outer loop through the attribute table
for each mrec in muidvtab
   muidcount=muidcount+1
   av.setstatus(muidcount/numrecs*100)
   muid=muidvtab.returnvalue(muidf,mrec)

   '---
   ' select current record in the muidvtab
   '**** because of the link, the corresponding
   ' records in the component table are also selected
****
   '---

   muidbit.clearall
   muidbit.set(mrec.clone)
```

```
   muidvtab.setselection(muidbit)
   muidvtab.updateselection

'--- inner loop through the component table

   for each crec in compvtab.getselection


       '--- get component name
       compname=compvtab.returnvalue(compnamef,crec)

       '--- get hypgrp name
       hydgrp=compvtab.returnvalue(hydgrpf,crec)
          if (hydgrp="C") then
              hydgrp="C*"
          end

       '--- get surftex name
       surftex=compvtab.returnvalue(surftexf,crec)


       '--- get comppct
       comppct=compvtab.returnvalue(comppctf,crec)

       '--- get muidseqnum
       muidseqnum=compvtab.returnvalue(muidseqnumf,crec)


       '--- set values in the output tablemu
   newrec=gavtab.addrecord
   gavtab.setvalue(outfields.get(0),newrec,muid)
   gavtab.setvalue(outfields.get(1),newrec,muidseqnum)
   gavtab.setvalue(outfields.get(2),newrec,compname)
   gavtab.setvalue(outfields.get(3),newrec,surftex)
   gavtab.setvalue(outfields.get(4),newrec,hydgrp)
   gavtab.setvalue(outfields.get(5),newrec,comppct)

   gamuidseqnum=gavtab.findfield("muidseqnum")
   gasurftex=gavtab.findfield("surftex")
   gaUSDA=gavtab.findfield("USDA")
   gahydgrp=gavtab.findfield("hydgrp")
   gacomppct=gavtab.findfield("comppct")


   gmuidseqnum = gavtab.returnvalue(gamuidseqnum, newrec)

'---convert USGS surface textures to USDA soil types---

   akey=gavtab.returnvalue(gasurftex,newrec)
   avalue=convdict.get(akey)
     if (avalue="0") then
```

```
       avalue=gavtab.returnvalue(gahydgrp,newrec)
      end
   gavtab.setvalue(outfields.get(6),newrec,avalue)

'---assigns G&A parameters from lookup table according to
USDA soil types---

    gcond = 0
    geffpor = 0
    gsuct = 0



  gcomppct = gavtab.returnvalue(gacomppct, newrec)
  gUSDA = gavtab.returnvalue(gaUSDA, newrec)

      for each lrec in lookupvtab

        lUSDA = lookupvtab.returnvalue(usda_code, lrec)


        if (gUSDA = lUSDA) then

          gcond = lookupvtab.returnvalue(condf, lrec)
          geffpor = lookupvtab.returnvalue(effporf, lrec)
          gsuct = lookupvtab.returnvalue(suctf, lrec)
          gavtab.setvalue(gcondf, newrec, gcond)
          gavtab.setvalue(geffporf, newrec, geffpor)
          gavtab.setvalue(gsuctf, newrec, gsuct)

          break
        end

     end




  gcond = 0
  geffpor = 0
  gsuct = 0

  '---calculate the holding capacity and enter it into
the USDA.dbf table---

      tempholdcap = 0
      holdcap = 0

   for each layrec in layervtab.getselection
```

```
      lmuidseqnum = layervtab.returnvalue(lmuidseqnumf,
layrec)

        if (gmuidseqnum = lmuidseqnum) then

          awcl = layervtab.returnvalue(awclf, layrec)
          awch = layervtab.returnvalue(awchf, layrec)
          laydepl = layervtab.returnvalue(laydeplf,
layrec)
          laydeph = layervtab.returnvalue(laydephf,
layrec)

          'conversion to SI units of millimeters
          'laydepl = laydepl * 25.4
          'laydeph = laydeph * 25.4

          templist.add({awcl,awch,laydepl,laydeph})

        end

   end

   for each laylist in templist

      awcl = laylist.get(0)
      awch = laylist.get(1)
      laydepl = laylist.get(2)
      laydeph = laylist.get(3)

      tempholdcap = ((awcl+awch)/2*(laydeph-
laydepl))*25.4 '---conversion to millimeters from inches
      holdcap = holdcap + tempholdcap


   end

   templist.empty
   gavtab.setvalue(outfields.get(10),newrec,holdcap)


   tempholdcap = 0
   holdcap = 0



   end   '--- end of loop through selected components



end     '--- end of muid loop
```

```
'---populate soils table

   pct_S = 0
   pct_LS = 0
   pct_SL = 0
   pct_L = 0
   pct_SIL = 0
   pct_SCL = 0
   pct_CL = 0
   pct_SICL = 0
   pct_SI = 0
   pct_SC = 0
   pct_SIC = 0
   pct_Clay = 0
   pct_A = 0
   pct_B = 0
   pct_C = 0
   pct_D = 0
   pct_W = 0
   pct = 0
   tempwgtcond = 0
   wgtcond = 0
   tempwgtsuct = 0
   wgtsuct = 0
   tempwgtchgmoist = 0
   wgtchgmoist = 0
   tempwgtinitloss = 0
   wgtinitloss = 0


  smapunitf=soilftab.findfield("muid")
if (smapunitf=nil) then
  msgbox.info("Cannot find mapunit field in the soils
table","")
  exit
end

gacompf=gavtab.findfield("comppct")
if (gacompf=nil) then
  msgbox.info("Cannot find component percentage field in
the output table","")
  exit
end


gamapunitf=gavtab.findfield("muid")
if (gamapunitf=nil) then
  msgbox.info("Cannot find mapunit field in the output
table","")
```

```
  exit
end

snumrecs = soilftab.getnumrecords
smuidcount = 0

   for each srec in soilftab

      smuidcount = smuidcount + 1
      av.setstatus(smuidcount/snumrecs*100)
      soilmapunit=soilftab.returnvalue(smapunitf,srec)

       for each orec in gavtab


        gamapunit=gavtab.returnvalue(gamapunitf,orec)

          if (soilmapunit=gamapunit) then

              pct = gavtab.returnvalue(gacompf, orec)
              soiltype = gavtab.returnvalue(gaUSDA, orec)
              cond = gavtab.returnvalue(gcondf, orec)
              suct = gavtab.returnvalue(gsuctf, orec)
              effpor = gavtab.returnvalue(geffporf, orec)
              holdcapacity = gavtab.returnvalue(holdcapf,
orec)

              complist.add({pct, cond, effpor, suct,
holdcapacity, soiltype})

          end

        end

'---calculate weighted parameters---

  for each valuelist in complist

     pct = valuelist.get(0)

     cond = valuelist.get(1)
     tempwgtcond = pct/100 * cond
     wgtcond = wgtcond + tempwgtcond

     effpor = valuelist.get(2)
     tempwgtchgmoist = pct/100 * effpor * (1 - s)
     wgtchgmoist = wgtchgmoist + tempwgtchgmoist

     suct = valuelist.get(3)
     tempwgtsuct = pct/100 * suct
     wgtsuct = wgtsuct + tempwgtsuct
```

```
     holdcapacity = valuelist.get(4)
     tempwgtinitloss = pct/100 * holdcapacity * (1 -
s)*2/10
     wgtinitloss = wgtinitloss + tempwgtinitloss

  '---calculate percentages of each soil type---

     soiltype = valuelist.get(5)
     if (soiltype = "S") then
       temppct_S = pct
       pct_S = pct_S + temppct_S
       temppct_S = 0

     elseif (soiltype = "LS") then
       temppct_LS = pct
       pct_LS = pct_LS + temppct_LS
       temppct_LS = 0

     elseif (soiltype = "SL") then
       temppct_SL = pct
       pct_SL = pct_SL + temppct_SL
       temppct_SL = 0

     elseif (soiltype = "L") then
       temppct_L = pct
       pct_L = pct_L + temppct_L
       temppct_L = 0

     elseif (soiltype = "SIL") then
       temppct_SIL = pct
       pct_SIL = pct_SIL + temppct_SIL
       temppct_SIL = 0

     elseif (soiltype = "SCL") then
       temppct_SCL = pct
       pct_SCL = pct_SCL + temppct_SCL
       temppct_SCL = 0

     elseif (soiltype = "CL") then
       temppct_CL = pct
       pct_CL = pct_CL + temppct_CL
       temppct_CL = 0

     elseif (soiltype = "SICL") then
       temppct_SICL = pct
       pct_SICL = pct_SICL + temppct_SICL
       temppct_SICL = 0

     elseif (soiltype = "SI") then
       temppct_SI = pct
```

```
      pct_SI = pct_SI + temppct_SI
      temppct_SI = 0

    elseif (soiltype = "SC") then
      temppct_SC = pct
      pct_SC = pct_SC + temppct_SC
      temppct_SC = 0

    elseif (soiltype = "SIC") then
      temppct_SIC = pct
      pct_SIC = pct_SIC + temppct_SIC
      temppct_SIC = 0

    elseif (soiltype = "C") then
      temppct_Clay = pct
      pct_Clay = pct_Clay + temppct_Clay
      temppct_Clay = 0

    elseif (soiltype = "A") then
      temppct_A = pct
      pct_A = pct_A + temppct_A
      temppct_A = 0

    elseif (soiltype = "B") then
      temppct_B = pct
      pct_B = pct_B + temppct_B
      temppct_B = 0

    elseif (soiltype = "C*") then
      temppct_C = pct
      pct_C = pct_C + temppct_C
      temppct_C = 0

    elseif (soiltype = "D") then
      temppct_D = pct
      pct_D = pct_D + temppct_D
      temppct_D = 0

    elseif (soiltype = "") then
      temppct_W = pct
      pct_W = pct_W + temppct_W
      temppct_W = 0

    end


  end

    complist.empty

    soilftab.setvalue(scondf, srec, wgtcond)
```

```
soilftab.setvalue(ssuctf, srec, wgtsuct)
soilftab.setvalue(schgmoistf, srec, wgtchgmoist)
soilftab.setvalue(sinitlossf, srec, wgtinitloss)
soilftab.setvalue(fPct_S, srec, Pct_S)
soilftab.setvalue(fPct_LS, srec, Pct_LS)
soilftab.setvalue(fPct_SL, srec, Pct_SL)
soilftab.setvalue(fPct_L, srec, Pct_L)
soilftab.setvalue(fPct_SIL, srec, Pct_SIL)
soilftab.setvalue(fPct_SCL, srec, Pct_SCL)
soilftab.setvalue(fPct_CL, srec, Pct_CL)
soilftab.setvalue(fPct_SICL, srec, Pct_SICL)
soilftab.setvalue(fPct_SI, srec, Pct_SI)
soilftab.setvalue(fPct_SC, srec, Pct_SC)
soilftab.setvalue(fPct_SIC, srec, Pct_SIC)
soilftab.setvalue(fPct_Clay, srec, Pct_Clay)
soilftab.setvalue(fPct_A, srec, Pct_A)
soilftab.setvalue(fPct_B, srec, Pct_B)
soilftab.setvalue(fPct_C, srec, Pct_C)
soilftab.setvalue(fPct_D, srec, Pct_D)
soilftab.setvalue(fPct_W, srec, Pct_W)
```

```
pct_S = 0
pct_LS = 0
pct_SL = 0
pct_L = 0
pct_SIL = 0
pct_SCL = 0
pct_CL = 0
pct_SICL = 0
pct_SI = 0
pct_SC = 0
pct_SIC = 0
pct_Clay = 0
pct_A = 0
pct_B = 0
pct_C = 0
pct_D = 0
pct_W = 0
pct = 0
wgtcond = 0
tempwgtcond = 0
wgtsuct = 0
tempwgtsuct = 0
wgtchgmoist = 0
tempwgtchgmoist = 0
wgtinitloss = 0
tempwgtinitloss = 0
```

```
   end

   '----------------------------------------
'---intersect watershed and soils themes---
'----------------------------------------

SRCTheme=soilTheme

interTheme=WatershedTheme


outFName=fn.merge(thedir.asstring,"soilintersect.shp")


'-------------------------------------------------------
'Set the variables
'-------------------------------------------------------

'outFName = av.GetProject.MakeFileName("theme", "shp")
'outFName = FileDialog.Put(outFName, "*.shp", "Output
Merged Shapefile")
if (outFName = Nil) then
  exit
end

shapeType = SRCTheme.GetFTab.FindField("Shape").GetType
'msgbox.info("type for intersect
theme="++shapetype.asstring,"")
'exit
if (shapeType = #FIELD_SHAPELINE) then
  outClass = POLYLINE
elseif (shapeType = #FIELD_SHAPEMULTIPOINT) then
  outClass = MULTIPOINT
elseif (shapeType = #FIELD_SHAPEPOINT) then
  outClass = POINT
elseif (shapeType = #FIELD_SHAPEPOLY) then
  outClass = POLYGON
else
  MsgBox.Error("Invalid shape field type.", "Merge
Themes")
  exit
end
'-------------------------------------------------------
'Set the variables
'-------------------------------------------------------
OutputFTab = FTab.MakeNew( outFName, outClass )
SRCfields = List.Make
InterFields = List.Make
for each f in SRCTheme.GetFTab.GetFields
  if (f.GetName = "Shape") then
    continue
```

```
  else
    fCopy = f.Clone
    SRCfields.Add(fCopy)
  end
end
interfieldlist=list.make
for each f in interTheme.GetFTab.GetFields
  if (f.GetName = "Shape") then
    continue
  else
    fCopy = f.Clone
    dCopy = F.Clone
    for each thefield in SRCfields
      'if the source field and the intersect field have
the same name
      'then truncate the name of the source field??
      if (thefield.getname = fcopy.getname) then
         if (fcopy.getname.count > 7) then

fcopy.setname((fcopy.getname.right(fcopy.getname.count -
1))+"b")
         else
           fcopy.setname(fcopy.getname+"b")
         end

      end
    end
    interFields.Add(fCopy)
    interfieldList.add({fCopy,dCopy})
  end
end
'-------------------------------------------------------
'add the fields to the output file
'-------------------------------------------------------
if (SRCFields.Count > 0) then
 OutputFTab.AddFields( SRCFields )
end
if (InterFields.Count > 0) then
 OutputFTab.AddFields( InterFields )
end
outshpfld = OutputFtab.findfield("Shape")
Theme1 = SRCTheme
ftab1=Theme1.getftab
shpfld1=ftab1.findfield("Shape")
therecs1 = ftab1.getselection
theoldsel = ftab1.getselection.clone

 if (therecs1.count=0) then
   therecs1=ftab1
 end
Theme2 = InterTheme
```

```
ftab2=Theme2.getFtab
shpfld2=ftab2.findfield("Shape")
therecs2 = ftab2.getselection
 if (therecs2.count=0) then
   therecs2=ftab2
 end
OutputFtab.seteditable(False)
OutputFtab.seteditable(True)
acount=0    'A counter variable
'-------------------------------------------------------
'Start processing each record in the selected overlay
polys
'-------------------------------------------------------
for each arec2 in therecs2     'loop through the overlay
polygons
 'get the polygon shape and select all records within
that shape
 theIntshape = ftab2.returnvalue(shpfld2,arec2)

 if (theView.getprojection.isNull) then
  Theme1.SelectbyShapes({theIntshape}, #VTAB_SELTYPE_NEW)
 else
  pshp=theIntShape.returnProjected(theView.getprojection)
  Theme1.SelectbyShapes({pshp}, #VTAB_SELTYPE_NEW)
 end

 'For each selected record
  recordcount = 0

  for each Selrec in ftab1.getselection   'loop through
selected
                                          'records in the
source polygons
    recordcount=recordcount+1
    av.showmsg("Splitting Shapes...")
    av.setstatus((recordCount / ftab1.getselection.count)
* 100)

    'Get the shape of the record

    SelectedShape = ftab1.returnvalue(shpfld1,Selrec)

    'If the output is a line
    '*** SMR
    'Ignoring the case of polylines for now
    '***
    if (outshpfld.getType = #FIELD_SHAPELINE) then

      'If the line is wholely within the polygin (no
intersection) then
       if (SelectedShape.iscontainedin(theIntShape)) then
```

```
         alineshp = SelectedShape

       else         'Else split the line using the
polygon
          alineshp =
SelectedShape.LineIntersection(theIntshape)
       end
        'Add the new record
         theoutrec=outputftab.addrecord
        'Set the shape value

outputFtab.SetValue(outshpfld,theoutrec,alineshp)
        'Set the field values
         for each afield in SRCfields
           oldfield=ftab1.findfield(afield.getname)
           oldvalue=ftab1.returnvalue(oldfield,selrec)
           outputftab.setValue(afield, theoutrec,
oldvalue)
         end
        'Set the field values
         for each afield in InterfieldList

oldfield=ftab2.findfield((afield.get(1).getname))
           oldvalue=ftab2.returnvalue(oldfield,arec2)
           outputftab.setValue(afield.get(0), theoutrec,
oldvalue)
         end



    else  'This enters into the polygon loop

      shpIntersect =
SelectedShape.ReturnIntersection(theIntshape)

      if (shpintersect.isempty) then
        continue
      end

      theoutrec=outputftab.addrecord

outputFtab.SetValue(outshpfld,theoutrec,shpIntersect)


      for each afield in SRCfields
        oldfield=ftab1.findfield((afield.getname))
        oldvalue=ftab1.returnvalue(oldfield,selrec)
        outputftab.setValue(afield, theoutrec, oldvalue)
      end
```

```
      for each afield in InterfieldList
        oldfield=ftab2.findfield((afield.get(1).getname))
        oldvalue=ftab2.returnvalue(oldfield,arec2)
        outputftab.setValue(afield.get(0), theoutrec,
oldvalue)
      end

    end 'if its poly or line

   av.purgeobjects

   end  'end for each selected record in the source theme


end 'ends the for each on arec1

Outputftab.seteditable(false)

ftab1.setselection(theoldsel)
ftab1.updateselection
if (MsgBox.YesNo("Add shapefile as theme to a view?",
  "Intersection", true).Not) then
  exit
end
' Create a list of views and allow the user to choose
which view to
' add the new theme to...
viewList = {}
for each d in av.GetProject.GetDocs
  if (d.Is(View)) then
    viewList.Add( d )
  end
end

' Include a choice for a new view...
viewList.Add("<New View>")

addToView = MsgBox.ListAsString( viewList,"Add Theme
to:", "Intersection" )

' Get the specified view, make the theme, and add it...
 if (addToView <> nil) then
  if (addToView = "<New View>") then
    addToView = View.Make
    addToView.GetWin.Open
  end

  mergeTheme = FTheme.Make( OutputFTab )
  addToView.AddTheme( mergeTheme )
  ' Bring the View to the front...
  addToView.GetWin.Activate
```

```
end
av.purgeobjects


'---fill in watershed table---

outputftab.seteditable(true)
oareaf = outputftab.findfield("area")
oshapef = outputftab.findfield("shape")
area=0
for each outrec in outputftab

  thepolygon = outputftab.returnvalue(oshapef, outrec)
  area = thepolygon.returnarea
  outputftab.setvalue(oareaf, outrec, area)

end

area=0


wshlist=list.make
alist=list.make
totalarea = 0
tempareacount = 0
areacount = 0
tempcondcount = 0
condcount = 0
tempsuctcount = 0
suctcount = 0
tempchgmoistcount = 0
chgmoistcount = 0
tempinitlosscount = 0
initlosscount = 0

ogridcodef=outputftab.findfield("gridcode")
ocondf=outputftab.findfield("cond")
osuctf=outputftab.findfield("Suct_Head")
ochgmoistf=outputftab.findfield("Chg_Moist")
oinitlossf=outputftab.findfield("Init_Loss")

'---make new fields in watershed table
addflist2=list.make


wcondf=wshftab.findfield("Cond")
if (wcondf = nil) then
addflist2.add(field.make("Cond", #field_decimal,8,3))
end

wsuctf=wshftab.findfield("Suct_Head")
```

```
if (wsuctf = nil) then
addflist2.add(field.make("Suct_Head",
#field_decimal,8,3))
end

wchgmoistf=wshftab.findfield("Chg_Moist")
if (wchgmoistf = nil) then
addflist2.add(field.make("Chg_Moist",
#field_decimal,8,3))
end

winitlossf=wshftab.findfield("Init_Loss")
if (winitlossf = nil) then
addflist2.add(field.make("Init_Loss",
#field_decimal,8,3))
end


if ((addflist2.count = 0).not) then
  wshftab.addfields(addflist2)
end
addflist2.empty

wcondf=wshftab.findfield("Cond")
wsuctf=wshftab.findfield("Suct_Head")
wchgmoistf=wshftab.findfield("Chg_Moist")
winitlossf=wshftab.findfield("Init_Loss")

for each wrec in wshftab

  wgridcode = wshftab.returnvalue(gridcodef, wrec)

    for each ourec in outputftab

      ogridcode = outputftab.returnvalue(ogridcodef,
ourec)

        if (wgridcode = ogridcode) then

            oarea = outputftab.returnvalue(oareaf, ourec)
            cond = outputftab.returnvalue(ocondf, ourec)
            suct = outputftab.returnvalue(osuctf, ourec)
            chgmoist = outputftab.returnvalue(ochgmoistf,
ourec)
            initloss = outputftab.returnvalue(oinitlossf,
ourec)

            wshlist.add({oarea, cond, suct, chgmoist,
initloss})
            totalarea=totalarea + oarea
         end
```

```
      end



      for each waterlist in wshlist

          oarea = waterlist.get(0)
          cond = waterlist.get(1)
          suct = waterlist.get(2)
          chgmoist = waterlist.get(3)
          initloss = waterlist.get(4)


          tempcondcount = cond*oarea/totalarea
          condcount = condcount + tempcondcount

          tempsuctcount = suct*oarea/totalarea
          suctcount = suctcount + tempsuctcount

          tempchgmoistcount = chgmoist*oarea/totalarea
          chgmoistcount = chgmoistcount +
tempchgmoistcount

          tempinitlosscount = initloss*oarea/totalarea
          initlosscount = initlosscount +
tempinitlosscount

       end

       wshlist.empty
       wshftab.setvalue(wcondf, wrec, condcount)
       wshftab.setvalue(wsuctf, wrec, suctcount)
       wshftab.setvalue(wchgmoistf, wrec, chgmoistcount)
       wshftab.setvalue(winitlossf, wrec, initlosscount)


      totalarea = 0
       tempcondcount = 0
       condcount = 0
       tempsuctcount = 0
       suctcount = 0
       tempchgmoistcount = 0
       chgmoistcount = 0
       tempinitlosscount = 0
       initlosscount = 0

   end
gavtab.seteditable(false)
wshftab.seteditable(false)
soilftab.seteditable(false)
```

References

Chow, V.T., D.R. Maidment, and L.W. Mays, *Applied Hydrology*, McGraw-Hill, Inc., New York, 1988.

Reed, S.M., Maidment D.R., and Patoux J., *Spatial Water Balance of Texas*, CRWR Online Report 97-1. Center for Research in Water Resources, University of Texas at Austin, February 1997

Reed, S.M. Ph.D., *Use of Digital Soil Maps in a Rainfall-Runoff Model*, CRWR Online Report 98-8. Center for Research in Water Resources, University of Texas at Austin, December 1998